Python OO course

Tool used: spyder 4.1.5 (inside Anaconda nav)

Python version: 3.8

# Classes and instances

Procedural languages focus on functions, OO programming stresses on objects.

An object is a **collection of data** (variables) and **methods** (functions) that act on those data. Similarly a **class is a blueprint for that object**.

We can think of a class as a sketch (**prototype**) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. “House” is the object.

As many houses can be made from a house’s blueprint, **we can create many objects from a class**. An **object** is also called an **instance of a class** and the process of creating this object is called **instantiation**.

Defining a class:

Class myNewClass:

Pass

(the “pass” keyword is used because python expects you to type something there, remember to use correct indentation)

A class creates a **new local namespace** where all its attributes (data or functions) are defined.

**Special attributes**: they begin with **double underscores”\_\_”:**

* “\_\_doc\_\_”: gives us the **docstring** of the class (used to document a specific segment of code, describe WHAT it does, not HOW). You can print it out by “print(<classname>.\_\_doc\_\_)”
* “\_\_init\_\_”: gets called whenever a new object of that class is **instantiated** (=constructor) and is used to initialize all the variables.

Defining a class with init:

class room:

"this is my room docspace"

def \_\_init\_\_(self, name, size):

self.name = name

self.size = size

myroom=room("My own room",125)

print(myroom.name)

print(myroom.size)

print(myroom.\_\_doc\_\_)

The “self” keyword should be used (same as ‘this’ in other languages) to follow conventions. It refers to the object itself. **The \_\_init\_\_ takes at least 1 argument “self”**

During instantiation, the self does not have to be defined, it happens automatically, “myroom” will be passed in as “self”.